Amendments to the Claims

1. (currently amended) A method to analyze a computer program that includes a plurality of blocks of code, the method comprising ~~the steps of~~:

~~executing said computer program~~ receiving a block of code to a code cache;

using a counter for tracking each time ~~one of said plurality of~~ said block[s] of code is executed on said code cache;

maintaining a counter cache for storing each said ~~plurality of counters~~ counter of said ~~plurality of~~ block[s] of code ~~that are most recently executed~~ while said block of code is stored on said code cache; and

maintaining a storage area for storing [a] each said ~~plurality of counters~~ counter of said ~~plurality of~~ block[s] of code ~~that are not most recently~~ previously executed on said code cache after said block of code is evicted from said code cache.

2. (currently amended) The method of Claim 1, further comprising the step of:

identifying when said ~~counter~~ code cache is full.

3. (canceled)

4. (currently amended) The method of Claim [3] 2, ~~wherein the copying steps further comprises the steps of~~ further comprising:

determining which ~~of~~ said ~~plurality of counters~~ counter of said ~~plurality of~~ block[s] of code ~~that are most recently executed~~ stored on said counter cache is least recently executed; and

~~copying~~ evicting said least recently executed block of code, related to said counter, from said ~~counter~~ code cache ~~to said storage area when said counter cache is full~~; and

copying said counter of said least recently executed block of code from said counter cache to said storage area when said least recently executed block of code related to said counter is evicted from said code cache.

5. (currently amended) The method of Claim [3] 1, wherein said receiving a block of code to a code cache further comprises ~~comprising the step of~~:

checking said ~~code cache~~ storage area to determine if [a] said block of code is being executed for other than the first time~~; and~~

loading [a] said counter associated with said block of code being executed for other than the first time, from said storage area into said counter cache; and

updating said counter associated with said block of code being executed for other than the first time.

6. (original) A system for analyzing a computer program that includes a plurality of blocks of code, comprising:

means for executing said computer program;

means for counting each time one of said plurality of blocks of code is executed;

means for maintaining a counter cache for storing said counting means of said plurality of blocks of code that are most recently executed; and

means for maintaining a storage area for storing said counting means of said plurality of blocks of code that are most recently executed.

7. (original) The system of Claim 6, further comprising:

means for identifying when said counter cache is full.

8. (currently amended) The system of Claim 7, further comprising:

means for copying ~~one of~~ said ~~plurality of~~ counting means of said plurality of blocks of code from said counter cache to said storage area when said counter cache is full. [.]

9. (currently amended) The system of Claim 8, wherein said identifying means further comprises:

means for determining which ~~of~~ said ~~plurality of~~ counting means of said plurality of blocks of code in said counter cache is least recently executed; and

means for copying said least recently executed block of code from said counter cache to said storage area when said counter cache is full.

10. (currently amended) The system of Claim 8, further comprising:

means for checking ~~said~~ a code cache to determine if a block of code is being executed for other than the first time; and

means for loading [a] said counting means associated with said block of code being executed for other than the first time, into said counter cache.

11. (currently amended) A computer readable medium having computer-readable program code embodied therein for causing a computer system to perform a method for analyzing a computer program that includes a plurality of blocks of code comprising:

~~logic for executing said computer program~~ receiving a block of code to a code cache;

~~logic for counting~~ utilizing a counter for tracking each time ~~one of~~ said ~~plurality of~~ block[s] of code is executed on said code cache;

~~logic~~ maintaining a counter cache for storing each said counter ~~counting logic~~ of said ~~plurality of~~ block[s] of code ~~that are most recently executed~~ while said block of code is stored on said code cache; and

~~logic~~ maintaining a storage area for storing each said counter ~~counting logic~~ of said ~~plurality of~~ block[s] of code previously executed on said code cache after said block of code is evicted from said code cache.

12. (currently amended) The computer readable medium of Claim 11, further comprising:

~~logic for~~ identifying when said code cache ~~most recently executed storing logic~~ is full.

13. (canceled)

14. (currently amended) The computer readable medium of Claim [13] 12, ~~wherein said logic for identifying~~ further comprises:

~~logic for~~ determining which [of] said ~~plurality of counting logic~~ counter of said ~~plurality of~~ block[s] of code in said ~~most recently executed storing logic~~ counter cache is least recently executed;

evicting said least recently executed block of code, related to said counter, from said code cache; and

~~logic for~~ copying said counter of said least recently executed block of code from said ~~most recently executed storing logic~~ counter cache to said storage ~~logic~~ area when said least recently executed block of code related to said counter is evicted from said code cache ~~most recently executed storing logic is full~~.

15. (currently amended) The computer readable medium of Claim 13, wherein said ~~logic for identifying~~ receiving a block of code to a code cache further comprises:

~~logic for~~ checking said ~~most recently executed storing logic~~ storage area to determine if [a] said block of code is being executed for other than the first time; ~~and~~

~~logic for~~ loading ~~a counting means~~ said counter associated with said block of code being executed for other than the first time, from said storage area into said ~~most recently executed storing logic~~ counter cache; and

updating said counter associated with said block of code being executed for other than the first time.

16. (currently amended) A system for analyzing a computer program that includes a plurality of blocks of code, the system comprising:

a counter that tracks each time ~~one of said plurality of~~ a specific block[s] of code is executed by a code cache;

a counter cache ~~that stores~~ for storing said ~~plurality of counters~~ counter of ~~said plurality of~~ a specific block[s] of code ~~that are most recently executed~~ while said specific block of code is stored on said code cache; and

a storage area for storing ~~that stores plurality of counters~~ said counter of ~~said plurality of~~ a specific block[s] of code ~~that are not most recently~~

previously executed on said code cache after said specific block of code is evicted from said code cache.

17. (currently amended) The system of Claim 16, further comprising:
logic that identifies when said ~~counter~~ code cache is full.

18. (canceled)

19. (currently amended) The system of Claim 17, wherein said logic determines which ~~of~~ said ~~plurality of counters~~ counter of ~~said plurality of~~ said specific block[s] of code stored on said [in] counter cache is least recently executed, evicting said least recently executed block of code related to said counter from said code cache, and copies ~~one of~~ said ~~plurality of counters~~ counter of ~~said plurality of~~ said specific block[s] of code from said counter cache to said storage area when said least recently executed specific block of code is evicted from said code cache ~~counter cache is full~~.

20. (currently amended) The system of Claim 17, wherein said logic checks said storage area ~~code cache~~ to determine if [a] said specific block of code is being executed for other than the first time, and loads a said counter associated with said specific block of code being executed for other than the first time, from said storage area into said counter cache, and updating said counter associated with said specific block of code being executed for other than the first time.